COSC460

RESEARCH PROJECT REPORT

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF CANTERBURY

APPLICATION DEVELOPMENT SYSTEMS

An alternative approach to Software Development

Stephen Hart

October 1982

Table of Contents

1. ABSTRACT 1

2. INTRODUCTION 2

3. WHAT IS AN APPLICATION DEVELOPMENT SYSTEM 5

4. APPLICATION DEVELOPMENT SYSTEM ENVIRONMENT 6

5. AIMS AND OBJECTIVES 7

6. WHAT IS AVAILABLE 8

7. THE ESSENTIAL FEATURES OF A DEVELOMENT SYSTEM ... 9

7.1 Program versus Parametized Generator 9

(i) Program Generators 9

(ii) Parameterized Generators 9

7.2 System Development Time 10

7.3 Ease of Maintenance 10

7.4 Prototyping 9

7.5 Exit Features 11

7.6 Efficient underlying Data Management Scheme . 11

7.7 Usability 11

7.8 Flexibility 12

8. APPLICATION SYSTEM REVIEW 13

9. FORMS 2 14

10. DATASTAR 15

11. The CONDOR Relational Data Base System 16

12. DBASE II Relational Data Base System 19

13. The FORMULA 22

14. SUMMARY 26

15. FUTURE DEVELOPMENTS 27

16. CONCLUSIONS 28

REFERENCES 29

APPENDICES

1. Example of FORMS 2 I

2. Case Study II

3. Test Application using CONDOR III

4. Test Application using DBASE II IV

5. Test Application using FORMULA V

ABSTRACT

Application Development Systems are now recognized as viable .ternatives in the area of software development. Although the lea of a high productivity approach to application development ; not new, it has only recently arrived as a force to help /ercome the problems associated with conventional programming :chniques.

This project is intended to examine these types of systems and) see how successfully they tackle the problems associated with)ftware productivity.

2. INTRODUCTION

The predominant feature of the computer industry over the last decade has been the substantial advances made in hardware technology. The net result of this has been the dramatic decrease in costs in most areas of computer hardware.

However there is one major area of concern that is associated with these advancements and that is the cost of software. The capacity to design and implement programming systems has not been able to match the impressive results achieved in hardware technology. This is due to a combination of many factors, notably :

(i) Systems design and programming is highly labour intensive and as such is directly related to the (ever increasing) cost of the people employed in the process.
(ii) The relative cost of software against hardware has also risen dramatically hence necessitating the reduction in software prices.
(iii) The inadequacy of existing development tools (i.e. procedural languages such as COBOL and PL/1) which are slow in terms of software productivity.

Allied to this is the situation where people without programming experience are completely familiar with an application but do not have the tools available to develop the application themselves. The time taken for an analyst to understand the requirements of the application and formulate a design specification adds substantially to the final cost of the application.

These factors combine to form the problem of the 'application backlog' - the demand for software is currently outstripping the available supply.

To attack these problems that result when conventional programming techniques are used, there exist the alternatives of :

(i) transfer as much responsibility as possible for the development of application(s) from the programmer to the end user.
(ii) increase programmer productivity.

To achieve the above goals a new range of tools and techniques have been developed that come under the general heading of 'Application Development Systems'.

These types of products are not the only methods used to increase software productivity (and hence reduce software costs). Other approaches that are used and are successful include :

(i) Improve current methods of system development by structured programming techniques (i.e. Jackson Structured Programming). However even if non structured programming techniques were previously used, there are figures that support a maximum possible increase in productivity of 25 percent[1].

(ii) Develop systems using the most powerful programming languages available. This includes the use of so called 'fourth generation languages' where the exact detail of what is required does not need to be specified. For example a typical statement such as ,

LIST STUDENT:NO STUDENT:NAME ADDRESS AGE

will produce a report. However it is the systems responsibility for retrieving and displaying the data.

(iii) Use software packages to provide the basis of the system. The classic example of this is IBM's Program for Airline Reservation System. This package has been successful because the requirements for handling airline reservations are virtually the same for all international airlines. Thus the basic aim of these packages is to avoid the problem of 're-inventing the wheel'. Unfortunately it is often the case that there is no wheel available to re-invent.

The use of packages may seem to be the most productive of the alternate solutions but there are generally severe limitations when using this type of product.

If the environment of an organization is such that its system requirements are static then the purchase of an appropriate package will require little modification to install and no modification in the future. In this type of situation the productivity of an application package is higher than that which could be achieved using a development system.

However it is often the case that system requirements vary from user to user and therefore when a package is purchased substantial modifications may be required in order for it to suit the users needs. Also a users requirements often change once a package has been installed. Maintaining the package can prove to be more expensive than the initial cost. Even worse is the situation where the vendors of the product refuse to perform modification on the system. In these cases the responsibility of maintaining the package lies with the user.

Thus in terms of maintenance and flexibility, (which are two qualities that should be provided by Development Systems) application packages are often not suitable.

With the use of Application Development Systems, end users are now given the ability to develop applications (although often in conjunction with an analyst). While some external design is still needed, it is nowhere as involved as that required with conventional programming techniques.

3. WHAT IS AN APPLICATION DEVELOPMENT SYSTEM

Any tool that aids software development is too broad an answer to this question as this can include anything from compilers, interpreters to application packages (i.e. IBM's Airline Reservation System).

A less abstract definition would be to consider these products as any file or data base oriented system that provides a series of high level facilities that aid system development. These 'high level facilities' are non procedural and as such allow applications to be generated quickly.

4. APPLICATION DEVELOPMENT SYSTEM ENVIRONMENT

The predominant area of use for these types of products is in the commercial world where the requirements are based on information retrieval plus the associated data manipulation (i.e. storage, modification and deletion of data). Moreover where the main method of viewing retrieved data is via reports (which is often the case in commercial institutions) then these products can really come into their own because of the facilities they provide for report generation.

These types of development tools are available for a majority of medium to large scale computer systems.

However with the drop in hardware prices there is now a wide range of microcomputers available to a new type of user. Software development at this level is subject to exactly the same productivity problems that are associated with mainframe systems.

There is a finite range of applications that can be developed with these kinds of development system. For example large number crunching pieces of software (such as missile guidance systems) cannot be expressed in terms of the facilities provided by development systems. Thus the only approach to develop this type of application is by using conventional programming techniques.

5. AIMS AND OBJECTIVES

The purpose of this project, with respect to these system development products, can be divided into the following sections :

(i) Find out what range (and type) of these products are available, especially at the lower end of the market.

(ii) Decide on what features and facilities these types of products need to provide in order to fulfill the goals that these tools are designed for. That is to say - 'What <u>is</u> required to' :

  (a) improve programmer productivity
  (b) shift emphasis of application development from programmers to end users

(iii) The previous objective will provide information about the qualities a system must possess in order for it to be a useful development aid. This will be used as the basis for the evaluation of the products found in (i).

(iv) Summarize the results obtained from (iii)

(v) Discuss what future developments can be expected with this type of product.

6. WHAT IS AVAILABLE

At the present time there is a wide range of products that come under the heading of Application Development Systems. There are two ends of the market that need to be considered with regard to these types of tools.

(i) Those systems developed for use in a mainframe environment. Almost all of these products are characterised by the fact that they are extremely dependent on the computer systems that they were developed for (so as to extract maximum efficiency) and hence not very portable. Examples include

(a) Burroughs LINC - Interfaces with Burroughs system software tools especially the Data Base Management System
(b) CBLs USER 11 - Heavily influenced by aspects of the RSTS operating system especially the file management system
(c) VISTA - A program generator developed specifically for PDP-11s.
(d) WANGs HAPAS and WASP - These products are aimed specifically at the cost accounting area [9]

Because these type of development systems have been covered in another part of the course it was not considered appropriate to discuss them again. This report will concentrate on those systems found in the following environment.

(ii) The systems that have been developed for use in the microcomputer environment. In this area there is a limited range of these types of products available. The scope of software considered was that which had been developed for the Z80 and CP/M based microcomputer. Because this type of system is very much dominant in the area of microcomputers, the products found in this environment can be considered to be representative of what is around.

The products that are available in the local market include :

(a) The CONDOR Relational Data Base System
(b) The DBASE II Assembly Language Relational Data Base System
(c) The Configurable Business System's FORMULA
(d) Micro Aps. SELECTOR IV

7. THE ESSENTIAL FEATURES REQUIRED FOR A DEVELOPMENT SYSTEM

In order to assess the power and capabilities of these development systems there are a variety of facilities that need to provided.

If a system is not characterised by the features specified then this limits its usefulness as a tool for system development. Following is a list of what is considered to be the essential features needed for a development system.

7.1 Program versus Parameterized Generators

Development systems can be divided into the following two categories :

(i) Program Generators

With this type of development system the interaction between the user and the system causes source code to be generated which forms either a part or the whole of the application.

(ii) Parameterized Generators

With these types of products it is not code that is generated but a set of parameters that are often referred to as a 'Data Dictionary'. So when creating a data base (or a file), information defining the structure of the records in the file is kept in the dictionary. Typical specifications of record fields include such information as :

(a) Field names, lengths and types
(b) Edit masks, default values and initial values
(c) Prompts and report headings

Thus when the various modules that make up the system are executed, they use the information in the dictionary to govern the action that will be taken. This type of generator is also known as an interpretive system in that it is not the code that is interpreted but the set of parameters that make up the dictionary.

The question of which approach is better is very much open to debate[8].

With program generators there is the question of how efficient is the code generated. Typically it is expected that 25 percent extra code is generated relative to hand coding [1]. The cost of recompilation whenever a change is made also needs to be considered. In the case of a system such as LINC changes in structure can result in the complete reorganization of the data base [8]. The time factor involved can make maintenance in a real time environment (e.g. a Banking System) impractical. Thus efficiency of recompilation is an important selection criteria

with this type of development system.

In interpretive systems these two questions do not arise because :

(a) No code is generated. If the modules are written in tightly optimized code then the efficiency (in terms of the resources required to run the application) is often better relative to a program generator.
(b) Changes in the underlying structure of the application implies that it is the parameters in the dictionary that are changed not the modules themselves. No recompilation of programs is required (as there are no programs) and all systems provide modules (or utilities) for handling changes in structure.

In terms of processing efficiency a system developed with with an Application Development System must be comparable in performance to the same system developed using conventional programming techniques.

If this is not the case and an application system requires, say, ten times the resources relative to a program based system then due to the overhead in terms of resources the application system may not be practical to use.

7.2 System Development Time

A measurement of the productivity of a system is the time it takes to develop applications relative to conventional programming techniques. Implicit in this is that the quicker an application can be developed, the cheaper its cost.

7.3 Ease of maintenance

A major justification of using an Application Development System is that changes can be easily coped with. This is especially true if programmers really do spend 80 percent of their programming time maintaining previously written software[7]. Also software maintenance is generally prompted by changes in requirements rather than problems associated with the application and so it is important that the lead time relative to a program based system be considerably reduced.

7.4 Prototyping

The ability to use an application generator as a tool for prototyping is also important.

Often an end user does not know exactly what he wants from a computer system. Thus the development of a prototype in conjunction with the end user allows for cheap modification of the prototype as opposed to being committed to the development of a system that may not be what the end user wants.

Typically reports, forms for data entry purposes and various other screen dialogues are developed by an analyst and an end user. A step by step refinement of the original design is made until a final design is reached.

The additional investment involved in the creation of the prototype is well justified by being able to have a relatively static final design.

7.5 Exit Features

Because of the non procedural language that these Application Development Systems are based on, not all applications can be generated in terms of the facilities provided by the system. Therefore in order to fill in the gaps it is essential to be able to incorporate modules written in procedural code.

7.6 Efficient underlying Data Management System

The management of data is a fundamental issue for any development system. There are two approaches to the overall control of data :

(i) File oriented
(ii) Data Base oriented

A data base system will be implicitly more flexible than a file system in that it is better able to handle the various access strategies that may be required to locate and retrieve data.

However with a file based system the logical relationships that exist between data is considerably more difficult to represent relative to a data base system.

7.7 Usability

To divert application development from programmers to end users implies that the system must provide a friendly user interface. Thus the system must be, to a large extent, self instructional. Ideally the methodology that a tool uses must be simple in order for end users to be able to develop applications with it.

The system must also have sufficient complexity to meet the needs of experienced programmers.

So to suit the requirements of both types of users a tiered approach should be used. That is there should be a subset of powerful non procedural commands that can be used by an end user while along side these there should be more procedural type commands that can be used by the experienced user.

7.8 Flexibility

Flexibility (in this context) refers to the ability of these development systems to provide the facilities for the the breadth of applications that users may wish to develop.

For example if a user wants to produce an accounting based application then the facilities provided should allow the development of data entry forms, menus, reports and the like.

8. APPLICATION DEVELOPMENT SYSTEM REVIEW

The following sections deal with the comparing of each system against the facilities (as outlined in the previous section) that a system must possess in order to meet the requirements of a good development system.

Information about the systems was gained through :

(i) Manuals and other forms of documentation that were available on the systems.
(ii) Development of the test application as detailed in Appendix[1].
(iii) Discussions with people who have had experience with these types of products.

Before considering the systems that have been categorized as Application Development Systems, there is also another range of tools that need to be mentioned.

These products are used in conjunction with procedural programming techniques and can be (very) useful if considered in the context of being aids to developing a system.

Following are two examples of these types of product that are used in the microcomputer environment.

9. FORMS 2

This system is not really a product designed for an end user but is a tool for an applications programmer.

The package allows the user to develop a form (on a terminal) corresponding to a particular layout wanted for data entry purposes. The tool is a program generator in that it uses the defined form to generate COBOL source code. The source code created will provide :

(i) Record definitions in the WORKING STORAGE SECTION that correspond to:

  (a) The background text as defined in the form.
  (b) The fields that will be used for data entry as defined in the form.

(ii) The ability to include code for the PROCEDURE DIVISION that will provide the basis of a program for handling data entry, modification and deletion. Given the previously defined form the user is expected to supply the code for handling the method of data entry and displaying the associated accompanying text.

(iii) The ability to include code for the ENVIRONMENT DIVISION that contains declarations that may be required for a file associated with the data fields of the form.

By incorporating the modules of code into a COBOL program then the problem of developing data entry routines is considerably simplified. In terms of productivity the time taken to develop the basis of a simple data entry form is in the order of 10-15 minutes. However the time taken to design the form as well as writing the equivalent code using conventional programming techniques, would be in the order of several hours.

10. DATASTAR

This product, like FORMS2, allows the user to develop a form that will define how and where data will be entered/displayed on a terminal screen.

There are two modules used to develop simple data entry routines when using this system :

(i) FORMGEN - Allows the user to define a form corresponding to the desired screen layout. As the system creates an indexed file, at least one of the data fields has to be selected for use as a key.
(ii) DATASTAR - Allows the user to enter, modify and review data for a particular file according to the form specified.

This system (unlike FORMS2) is parameter driven in that the module DATASTAR uses the set of parameters created by FORMGEN to govern such actions as :

(i) How and where the form is displayed
(ii) Simple edit checks
(iii) Duplicate key handling

The data (and index) file that are created can be used by a number of procedural languages (such as COBOL and BASIC).

If this product was used as a basis for the file maintenance part of a system then the productivity of this system would be very good. The time taken to develop a (simple) data entry routine was in the order of 15 to 20 minutes. The corresponding time taken using conventional programming techniques would be at least a day. A COBOL program to provide the facilities that DATASTAR provides, (although being more flexible), would require in the area of a 1000 lines of code.

11. The CONDOR Relational Data Base System

This system is designed for developing applications using any 'Z80 based CP/M' type of microcomputer.

The range of facilities provided by this system allow for :

(i) Data entry capabilities
(ii) File processing capabilities
(iii) Reporting features

The basic approach to developing applications in using this system involves creating command files which consist of a sequence of the non procedural commands that the system provides.

Menus can be created which allow options to be selected that trigger off the execution of these command files.

The power of the commands make this system (relatively) easy to understand for end users. However at this level of abstraction, experienced users will find that the facilities provided severely limit the range of applications that can be produced.

In accordance with the requirements of a development system, as discussed in Section 7 the following observations can be made about the system.

11.1 Parameterized versus Program Generator

CONDOR is in fact a parameter driven system in that a dictionary is associated with each file (known as a data base in CONDOR) created by the user. The dictionary contains the attributes of the file and it is these attributes which are accessed by each particular module at execution time.

There is also a form definition file associated with the dictionary and the data file. The information kept in this file is the definition of a form that is used to provide the interface between the user and the data file.

11.2 System Development Time

Getting an application up and running can be achieved very quickly. The predominant factors that allow this are :

(i) The ability to let the user develop forms corresponding to how and where data will be entered for a particular file.
(ii) The very high level facilities that let the user perform complex operations in terms of a single command.

The test application that was selected took approximately 4 days to develop. This compares favourably with a time factor in the order of weeks as would have been required if conventional programming techniques were used. So in the area of programmer productivity CONDOR is, for some applications, considerably more powerful than conventional programming techniques

11.3 Ease of Maintenance

The ability to make changes to an application is helped considerably by the self documenting nature of the commands that make up the system.

Any restructuring actions that may be required can also be handled in terms of the facilities provided.

11.4 Prototyping

Because this system is only useful for developing small applications, the ability to prototype an application is not required.

However if using CONDOR as a prototyping tool then this system is only useful in certain areas. In modelling a large scale application then CONDOR is good for :

(i) Developing various menu oriented dialogues that may be required
(ii) Defining screen layouts for data entry purposes

With reports, the area where most changes and ad hoc requests occur, the facilities provided only allow a limited form of report generation. The type of report that can be produced is not elegant and would not satisfy the needs of many users.

11.5 Exit features

One of the failings of this system is its inability to elegantly interface external programs to the system.

Because of its file structure, any program wanting to read (or write) from (to) a file has to firstly create a temporary ASCII version of the file to work with.

11.6 Efficient underlying Data Base System

The major disadvantage with CONDOR comes under this category. All table driven systems are inherently slow even with some form of indexing. However CONDOR provides no indexing and the only method of data access is serial.

In terms of response, this approach to data retrieval is extremely poor and it is this aspect which makes CONDOR unacceptable as a good Application Development System.

The method of data access is not specified in the manual but is implied because it recommends that transaction processing should be done in batches and that sorts on relevant fields be done whenever a data base operation such as JOIN or PROJECT is done.

Any application that is based on on-line transaction processing is not feasible due to the unacceptable response times the system provides. The initial version of the test application was based on the above approach. Even with a file of 50 records the response times were in the order of 7 seconds.

11.7 Usability

The system has been developed for users with limited computing experienced rather than as a tool for experienced programmers.

This is reflected in the basic set of commands that need to be mastered in order to create a data base and perform subsequent data manipulation on the file.

Thus developing a simple application in terms of the facilities provided by the system is very straightforward. However for anything approaching a complex application, it is simply not feasible to develop it within the confines of this system.

11.8 Flexibility

In terms of flexibility CONDOR is not to good. While it is alright to develop simple applications with it, for a more complex application concessions often have to be made.

With this system the designers have sacrificed flexibility for usability. While the high level commands are very powerful they take too much control out of the users hands.

The method of data management also inhibits the flexibility of the system. Any application that requires some form of random access will be subject to slow response times.

For the test application that was selected it was not possible to map the original design of the application to this system in terms of the facilities provided by the system.

12. The DBASE II Assembly Language Relational Data Base System

This system is basically a relational data base management system that provides a user with the facilities to :

(i) Develop a data base system
(ii) Perform processing operations on the data base system
(iii) Generate reports from the information in the data base
(iv) Define command modules which use both procedural and non procedural commands. These allow the development of complex applications that are comparable to conventional programming techniques.

This system has, like CONDOR, been developed for use on any Z80 based microprocessor system that has an operating system such as CP/M.

In terms of the essential characteristics that an Application Development System must provide, it can be seen that DBASE II has the following features :

12.1 Parameterized versus Program generator

Like CONDOR this system is parameter driven. Details such as the names, lengths and types of various fields are kept as header records in the data file (known as a data base in DBASE II). All interaction between the user and the files he has created is in a fixed format.

12.2 System Development Time

The time taken to develop the test application was in the order of 9 days. However this was entirely due to the development of the application by using the procedural capabilities that the system provides as opposed to the using the less flexible non procedural commands that are also available.

The productivity that can be gained from using this tool is very high. In any non trivial application the user can develop command files in terms of both procedural and non procedural commands. When a command file developed in this way is compared with a procedural program performing the same task then in terms of development time DBASE II is considerably more productive.

12.3 Ease of Maintenance

Under this heading, the reorganizing and restructuring facilities provided enable the user to perform aspects of data base modification without very much trouble.

If an application was developed in terms of the procedural commands that the system provides then maintenance of the application is certainly more difficult than if non procedural

commands had been used. In fact maintenance in this case is subject to virtually the same constraints as a typical programming system.

12.4 Prototyping

In the microcomputer environment the prototyping of an application is not as important as in a mainframe environment. This is because the complexity of application for microcomputers is very minor relative to a large scale application. Thus the ability to envisage a final design will be considerably easier for an application on a microcomputer than an application on a mainframe.

The facilities provided by DBASE II do not allow some of the more useful aspects of a prototype to be developed when building a model of an application.

An analyst in conjunction with an end user can quickly do things such as create data bases and generate reports. However there is no quick way of developing such things as screen menus, complex data entry forms and the like. The only way of handling those type of requirements is by using the non procedural commands that are provided and this would be beyond the ability of an end user.

12.5 Exit features

DBASE II does not provide any escape features. However because the system provides a variety of procedural commands as well as the non procedural commands almost any application can be expressed in terms of the facilities provided by the system.

12.6 Efficient underlying Data Management System

DBASE II is based around a relational data base management system. The standard organization of the data files is serial as in CONDOR (see 11.6) and this approach will be inherently slow.

However there exists an alternative method of accessing data and that is by the creation of an index file associated with a particular data file. The index file is represented in terms of a B*-tree and thus with a file that is n records in size, access to a random record will be in the order of log2n accesses. Thus the use of an index file in conjunction with the data file will greatly improve the data management scheme in terms of both efficiency and speed of access.

DBASE II in fact provides for both multiple index keys and multiple index files. However only one index file in conjunction with the data file can be used at once.

12.7 Usability

From an end users point of view this system would be easier to use than CONDOR in that equivalent non procedural commands are more powerful than those supplied by CONDOR. For example if a user wants to update a record using CONDOR he has to formulate a condition in order to locate the desired record. In DBASE II the user is prompted with the fixed screen layout of the record and allowed to 'fill in the blanks' for as many fields as he wants so to narrow the search for the record.

However the procedural commands available (which allow the development of command files) virtually require normal programming techniques that could only be mastered by a programmer or someone who had more than a short period of training with the system.

12.8 Flexibility

When developing the test application no concessions had to be made in order to map the original design to this system. The only reason for this was the use of the facilities provided by the procedural language. If a user was restricted to the non procedural commands then it would not be possible to develop anything other than the simplest of applications.

13. The FORMULA

This system has been designed to allow end users to develop business oriented applications on any 'Z80 based CP/M' microcomputer.

In order to allow users (who have had no prior experience in programming) to develop applications a menu approach is used. With this system a user is provided with a special menu (called the 'Designer Menu') which contains all the necessary options required to build an application.

The system is file oriented and provides for three basic functions that can be performed on files :

(i) Data entry - To provide users with the ability to perform addition, modification and deletion on a data file.

(ii) File processing - Batch oriented updating of data files is provided by this system. A file can be updated from another file based on some relationship between the two files.

(iii) Report Generator - Basic types of reports (along the line of the other two systems) can be produced as well as word processing types of activities (e.g. letters where names and addresses are obtained from file look ups).

Under the categories described in section 7 it is possible to make the following observations :

13.1 Parameterized versus Program Generator

Like the previous two systems, this is a parameter driven system. The file creation module is used to create a set of parameters based on a users response to questions concerning field definitions. These parameters are kept in a series of header records at the beginning of the data file.

13.2 System Development Time

The menu approach to system development allows applications to be developed very quickly (relative to conventional techniques).

For example the time to develop the basis of the test application was less than a day.

Each menu option that exists in the Designer Menu has assocaited with it a set of high powered non procedural facilities. It is these facilities provided by the system which result in the high productivity gains that can be achieved.

13.3 Ease of Maintenance

Because the FORMULA is completely menu driven then to accommodate for a change in requirements revolves around the type of modification required :

(i) Changes in logic simply involve changing a menu option and the commands associated with that option.
(ii) Changes in file structure involves defining a new file (with the file creation option in the Designer Menu) and copying the desired fields of the old file to the new file.

The menu approach is self documenting and therefore it is straightforward for another person to 'pick up the pieces' from where an application was left at by another user.

13.4 Prototyping

As a prototyping tool, the FORMULA is quite useful. Time taken to develop screen dialogues, data files and reports is very short.

Because maintenance on the system can be performed very quickly, the step by step refinement process required to arrive at a final design can also be easily achieved.

13.5 Exit Features

The designers of this product have recognized the fact that not all applications can be developed using the facilities provided by the system.

To overcome this a user is able to chain to a menu option (which can consist of up to 10 sequentially executed commands) modules written in a suitable procedural language.

The inclusion of modules of custom code into an application can be used to overcome such problems as :

(i) Developing screen forms that only deal with part of a data file
(ii) Incorporating security checks
(iii) Developing different types of screen dialogues from what the system provides

13.6 Efficient underlying Data Management System

The FORMULA is based around the concept of a file oriented system. With this system there are two types of files for organizing data:

(i) Master Files - Are indexed and based on one primary key, namely the first field specified in the record.
(ii) Transaction Files - Are purely serial in organization and the purpose of these files is when random access is not required or to update master files.

Given that this system is developed for business oriented applications the idea in (ii) works quite well because the concept of a master file having an associated transaction file occurs quite frequently in this type of environment (e.g. a General ledger and transaction file)

The ability to have a file based on only one key is a limiting factor. The only alternative method of changing the logical view of the file is by the creation of a pointer file according to a set of specified conditions.

However this file is not maintained so whenever records are added or deleted from the file, the pointer file will not reflect the change. A typical case of where an alternate view of data is required is in the production of reports. For example a student file may be keyed on student number but a report is required for students in alphabetical order. The creation of a pointer file, while slow for larger files, will allow this type of report to be generated.

13.7 Usability

As noted this system is menu driven which is probably the best way to provide the man-machine interface for inexperienced computer users. By making the system virtually self documenting it is possible for end users to get a simple system going without much effort. Allied to this is the reference manual which, as well as introducing a user to basic concepts such as files and the like, provides a series of examples which can be worked through in a step by step fashion.

13.8 Flexibility

As with the CONDOR system the designers of the FORMULA sacrificed flexibility for usability. The ability to develop applications in terms of menus is good for end users but not for experienced programmers.

The scope of applications that can be developed with this system is limited to accounting related systems.

The file oriented approach to data management means that for any application where the logical relationship between data is complex, the file representation will not be as efficient (in terms of data manipulation) as a data base system.

14. SUMMARY

The application software considered contained many shortcomings. None of the products fulfilled the specified requirements for a good development system.

For instance in the area of data management, which in the case of microcomputers is the most important factor in processing efficiency, only DBASE II provided a scheme that was able to adequately handle the variety of data manipulation operations that could be expected to occur in an application.

Because no system was fully able to meet the specifications of section 7, then none of the products are appropriate for developing all the differing types of applications that can occur in the microcomputer environment.

Thus when an application is to be developed it is important that the tool that is selected is appropriate for the task.

For example if an application was required where ad hoc reports were often needed then a system with a good report generator such as 'the FORMULA' should be selected as opposed to a system such as CONDOR.

Most of these products do not provide any procedural capabilities. This has made these types of systems inflexible from a programmers point of view. Thus these types of product that are available in the microcomputer environment help to reduce the application backlog by increasing the software productivity via the end users. By allowing the user to take responsibility for application development, this will allow programmers to invest more of their time in reducing the application backlog.

This situation is reflected in places like MicroAge International, a company that specialises in software development. To date their only involvement with these types of products is as vendors, not as users themselves.

However in the mainframe environment, where application development systems are better geared towards programmer productivity, there are companies who spend a lot of time developing applications with application development systems (e.g. CBL who use USER 11).

15. FUTURE DEVELOPMENTS

At the current state of development, packaged applications and Application Development Systems are to a large extent complimentary.

Application Packages are moving in the direction of application generators by being more flexibly parameterized and including such things as special purpose report generators.

The key concept here is in the parameterization of packages [1]. By designing packages with more parameters which can be selected and varied the package can fit into an organization with as little modification as possible.

On the other hand application generators will (continue to) move in the direction of packages until they are able to match the custom built facilities provided by application packages.

Another theory that has been put forward[1] is that in the future packages will be generated by development systems so that an end user can modify them. The ability to modify a package to suit a users requirements would be very handy.

Considerable work has been put into the efficiency problems associated with these types of products. An example of this is the work done by the The Systems Group, an American corporation who have developed a new type of development system called Business Express[12]. There method has been to improve on the inflexible approach of having both the operating system and the development system as two separate entities. To achieve a high performance factor on a microcomputer they have incorporated the two items into one entity. As well as this they have taken into consideration such things as disk buffering and the use of resident re-entrant code in order to optimize other performance areas.

With this radical approach they have claimed to be able to support 10 terminal users without any degredation.

The important thing here is that sophisticated software is being developed for microcomputers and in the future a more powerful range of products will become available for these types of systems.

a 16. CONCLUSIONS

Application Development Systems are the most important tool in improving software productivity in large scale systems. This is substantiated by the huge gains in productivity obtained from these tools relative to any other method[1].

In the microcomputer environment these types of products are nowhere near as common or sophisticated as those found on mainframes.

However the systems in this environment do increase software productivity and hence reduce the application backlog. The approach most of these systems use is to design these products for end users, not for programmers per se. This is reflected in the high powered non procedural facilities offerred and the lack of procedural facilities.

So these tools do not as yet reduce the application backlog via improving programmer productivity.

Things are only now beginning to change in this area. With the advent of products such as Business Express and The Last One, the sophistication these products offer will be considerably higher than what is now available. With these types of tools it can be expected that improvements in programmer productivity will result as well as end user productivity.

REFERENCES

[1] MARTIN J, 'Application Development without Programmers'

[2] CONDOR Series 20 r/DBMS: Users Operation Manual, CONDOR COMPUTER CORPORATION

[3] DBASE II - Assembly language Realational System, ASHTON-TATE

[4] THE FORMULA - Users Guide and Reference Manual, DYNAMIC MICROPROCESSOR ASSOCIATED

[5] DEARNLY PA, 'Software Development for Microcomputer Data Processing Systems', British Computer Journal, VOL 25 NO. 2 1982

[6] MITCHELL KI, 'LINC - An independent view', INTERFACE, April 1982

[7] JERVIS A, 'Is LINC the answer ?', INTERFACE, June 1982

[8] JERVIS A, 'System Development Products, with special reference to LINC', INTERFACE, July 1982

[9] GALLOWAY C, 'Sting in battle for Accountants Market', INTERFACE, April 1982

[10] DATASTAR - Users guide and reference manual, DIGITAL RESEARCH

[11] DUNN L, 'Business Express: An integrated application development system and multi user operating system', LAMANTIA MARKETING COMMUNICATIONS INCORPORATED

[12] BUSINESS EXPRESS - Structure, Design and implementation methods, SYSTEMS GROUP

Appendix 1

The following listings show the results that can be achieved by using the program generator FORMS 2 :

* Background text of the created form
* Data entry points on the created form
* Code generated for Working Storage Section based on the created form
* Code for the Procedure Division that provides a partially completed file maintenance routine

CHRISTCHURCH TEACHERS COLLEGE -- TRANSACTION FILE MAINTENANCE

TRANSACTION RECORD DETAILS

DATE :
CODE :
NUMBER :
DETAILS :
TYPE [D/C] :
AMOUNT :

IS DATA ENTERED CORRECTLY ? [Y/N]

999999
9999
99999
XXXXXXXXXXXXXXXXXXXXXXXXXX
X
9999999.99

X

```
 01   TFLSCR-00
      03 TFLSCR-00-0001    PIC X(0002) VALUE "U7".
      03 FILLER            PIC X(0086).
      03 TFLSCR-00-0002    PIC X(0061) VALUE "CHRISTCHURCH TEACHERS
-     "COLLEGE -- TRANSACTION FILE MAINTENANCE".
      03 FILLER            PIC X(0012).
      03 TFLSCR-00-0003    PIC X(0076) VALUE "======================
-     "==========================================================".
      03 FILLER            PIC X(0251).
      03 TFLSCR-00-0004    PIC X(0026) VALUE "TRANSACTION RECORD DET
-     "AILS".
      03 FILLER            PIC X(0139).
      03 TFLSCR-00-0005    PIC X(0004) VALUE "DATE".
      03 FILLER            PIC X(0008).
      03 TFLSCR-00-0006    PIC X(0001) VALUE ":".
      03 FILLER            PIC X(0067).
      03 TFLSCR-00-0007    PIC X(0004) VALUE "CODE".
      03 FILLER            PIC X(0008).
      03 TFLSCR-00-0008    PIC X(0001) VALUE ":".
      03 FILLER            PIC X(0067).
      03 TFLSCR-00-0009    PIC X(0006) VALUE "NUMBER".
      03 FILLER            PIC X(0006).
      03 TFLSCR-00-0010    PIC X(0001) VALUE ":".
      03 FILLER            PIC X(0067).
      03 TFLSCR-00-0011    PIC X(0007) VALUE "DETAILS".
      03 FILLER            PIC X(0005).
      03 TFLSCR-00-0012    PIC X(0001) VALUE ":".
      03 FILLER            PIC X(0067).
      03 TFLSCR-00-0013    PIC X(0010) VALUE "TYPE [D/C]".
      03 FILLER            PIC X(0002).
      03 TFLSCR-00-0014    PIC X(0001) VALUE ":".
      03 FILLER            PIC X(0067).
      03 TFLSCR-00-0015    PIC X(0006) VALUE "AMOUNT".
      03 FILLER            PIC X(0006).
      03 TFLSCR-00-0016    PIC X(0001) VALUE ":".
      03 FILLER            PIC X(0223).
      03 TFLSCR-00-0017    PIC X(0033) VALUE "IS DATA ENTERED CORREC
-     "TLY ? [Y/N]".
 01   TFLSCR-01    REDEFINES TFLSCR-00    .
      03 FILLER            PIC X(0001).
      03 TFLSCR-01-0001    PIC 9(0001).
      03 FILLER            PIC X(0665).
      03 TFLSCR-01-0002    PIC 9(0006).
      03 FILLER            PIC X(0074).
      03 TFLSCR-01-0003    PIC 9(0004).
      03 FILLER            PIC X(0076).
      03 TFLSCR-01-0004    PIC 9(0005).
      03 FILLER            PIC X(0075).
      03 TFLSCR-01-0005    PIC X(0024).
      03 FILLER            PIC X(0056).
      03 TFLSCR-01-0006    PIC X(0001).
      03 FILLER            PIC X(0079).
      03 TFLSCR-01-0007    PIC 9999999.99.
      03 FILLER            PIC X(0247).
      03 TFLSCR-01-0008    PIC X(0001).
```

```
01  CURSOR-POSITION        PIC 9(4) VALUE ZERO.
01  INDICATORS.
    02 CURRENT-RECORD      PIC X(3) VALUE "NO".
    02 END-OF-FILE         PIC X(3) VALUE "NO".
    02 KEY-CHANGED         PIC X(3).
    02 DATA-CHANGED        PIC X(3).
01  COMMENT                PIC X(50) VALUE SPACE.

PROCEDURE DIVISION.
START-UP.
    DISPLAY SPACE.
    PERFORM SET-UP-FILE-NAME.
    OPEN I-O INDEXED-FILE.
    MOVE SPACE TO INDEXED-RECORD.
    MOVE SPACE TO SAVED-RECORD.
    PERFORM DISPLAY-FORM.
    GO TO NO-CURRENT-RECORD.
ACCEPT-FROM-SCREEN.
    MOVE START-OF-DATA TO CURSOR-POSITION.
    PERFORM ACCEPT-RECORD THRU SET-UP-RECORD.
    IF RECORD-KEY NOT = SAVED-KEY
        MOVE "YES" TO KEY-CHANGED
    ELSE MOVE "NO" TO KEY-CHANGED.
    IF RECORD-DATA NOT = SAVED-DATA
        MOVE "YES" TO DATA-CHANGED
    ELSE MOVE "NO" TO DATA-CHANGED.
    MOVE INDEXED-RECORD TO SAVED-RECORD.
    IF KEY-CHANGED = "YES" GO TO SEE-IF-RECORD-EXISTS.
    IF DATA-CHANGED = "YES" GO TO UPDATE-RECORD.
    IF END-OF-FILE = "YES" GO TO CLOSE-DOWN.
    IF CURRENT-RECORD = "NO" GO TO SEE-IF-ANY-MORE-RECORDS.
    IF CURRENT-RECORD = "DUP" GO TO REPLACE-RECORD.
    IF CURSOR-POSITION NOT = START-OF-KEY
                            GO TO SEE-IF-ANY-MORE-RECORDS.
*DELETE-RECORD.
    DELETE INDEXED-FILE.
    MOVE "RECORD DELETED" TO COMMENT.
    MOVE SAVED-KEY TO RECORD-KEY.
    GO TO NO-CURRENT-RECORD.
SEE-IF-ANY-MORE-RECORDS.
    START INDEXED-FILE KEY > RECORD-KEY
          INVALID GO TO END-OF-FILE-REACHED.
    READ INDEXED-FILE NEXT.
    GO TO CLEAR-COMMENT.
END-OF-FILE-REACHED.
    MOVE "END OF FILE REACHED - RETURN WILL TERMINATE"
        TO COMMENT.
    MOVE "YES" TO END-OF-FILE.
    MOVE SPACE TO RECORD-KEY.
    GO TO NO-CURRENT-RECORD.
REPLACE-RECORD.
    IF CURSOR-POSITION = END-OF-DATA GO TO AMEND-RECORD.
    IF CURSOR-POSITION = START-OF-KEY GO TO AMEND-RECORD.
    MOVE "YES" TO CURRENT-RECORD.
    GO TO SEE-IF-RECORD-EXISTS.
UPDATE-RECORD.
    IF CURRENT-RECORD NOT = "YES" GO TO SEE-IF-RECORD-EXISTS.
AMEND-RECORD.
    REWRITE INDEXED-RECORD.
    IF CURRENT-RECORD = "DUP"
        MOVE "RECORD REPLACED" TO COMMENT
    ELSE MOVE "RECORD AMENDED" TO COMMENT.
```

```
           GO TO SET-CURRENT-RECORD.
       SEE-IF-RECORD-EXISTS.
           READ INDEXED-FILE INVALID GO TO SEE-IF-NEW-RECORD.
           IF DATA-CHANGED = "NO" GO TO CLEAR-COMMENT.
       RECORD-ALREADY-EXISTS.
           MOVE "RECORD ALREADY EXISTS WITH THIS KEY" TO COMMENT.
           MOVE SAVED-RECORD TO INDEXED-RECORD.
           MOVE "DUP" TO CURRENT-RECORD.
           GO TO NOT-END-OF-FILE.
       CLEAR-COMMENT.
           IF CURRENT-RECORD = "DUP" GO TO RECORD-ALREADY-EXISTS.
           IF CURSOR-POSITION = END-OF-DATA
                                     GO TO RECORD-ALREADY-EXISTS.
           IF CURSOR-POSITION = START-OF-KEY
                                     GO TO RECORD-ALREADY-EXISTS.
           MOVE SPACE TO COMMENT.
           GO TO SET-CURRENT-RECORD.
       SEE-IF-NEW-RECORD.
           MOVE SAVED-RECORD TO INDEXED-RECORD.
           IF DATA-CHANGED = "YES" GO TO WRITE-NEW-RECORD.
           IF CURRENT-RECORD = "DUP" GO TO WRITE-NEW-RECORD.
           IF CURSOR-POSITION = END-OF-DATA GO TO WRITE-NEW-RECORD.
           IF CURSOR-POSITION = START-OF-KEY GO TO WRITE-NEW-RECORD.
           MOVE "RECORD NOT FOUND" TO COMMENT.
           MOVE "NO" TO END-OF-FILE.
       NO-CURRENT-RECORD.
           MOVE SPACE TO RECORD-DATA.
           PERFORM SET-UP-SCREEN.
           PERFORM SET-UP-RECORD.
           MOVE "NO" TO CURRENT-RECORD.
           GO TO SAVE-RECORD-AREA.
       WRITE-NEW-RECORD.
           WRITE INDEXED-RECORD.
           MOVE "NEW RECORD WRITTEN" TO COMMENT.
       SET-CURRENT-RECORD.
           MOVE "YES" TO CURRENT-RECORD.
       NOT-END-OF-FILE.
           MOVE "NO" TO END-OF-FILE.
       SAVE-RECORD-AREA.
           MOVE INDEXED-RECORD TO SAVED-RECORD.
           PERFORM SET-UP-SCREEN THRU DISPLAY-RECORD.
           DISPLAY COMMENT AT COMMENT-POSITION.
           GO TO ACCEPT-FROM-SCREEN.
       CLOSE-DOWN.
           CLOSE INDEXED-FILE.
           DISPLAY SPACE.
           DISPLAY "RUN TERMINATED" AT 1030.
           STOP RUN.
      *
       SET-UP-FILE-NAME.
```

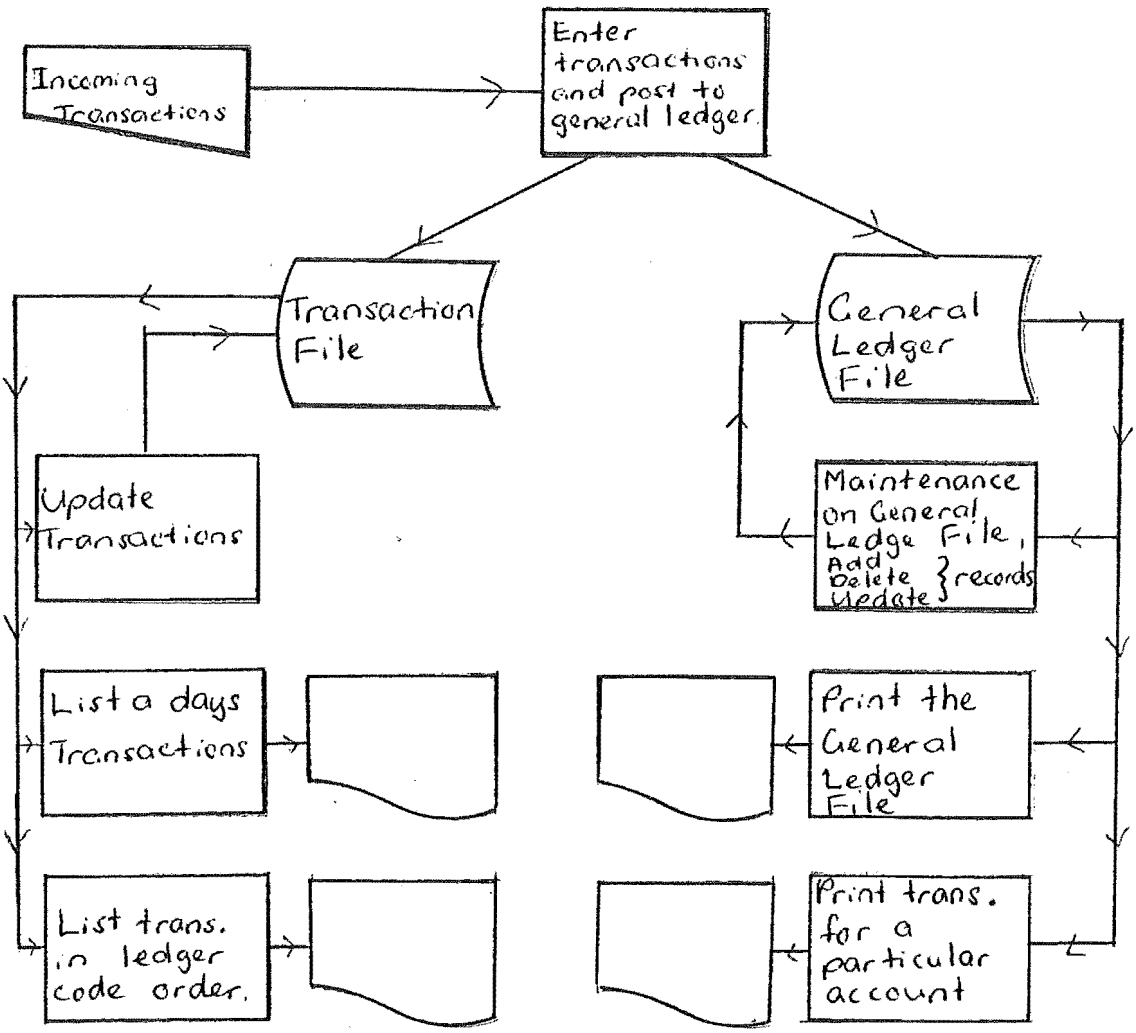
Case Study

In order to gain an appreciation of the performance capabilities of these development systems, an application was selected for the purpose of a case study.

The application chosen was a 'Printery Control System' and is basically a subset of an accounting system.

The original system consisted of a suite of 15 menu driven COBOL programs. Time of development was in the order of 4 - 5 weeks given that the specification of the system was already known.

Printery System Design



Appendix 3

The CONDOR Development System

Following are some examples of what can be produced using the facilities provided by the system.

* The form developed for the General Ledger File
* The attributes of the file as specified by the dictionary associated with the file
* A listing of the General Ledger
* One of the menus for the system
* A command file consisting of a sequence of non procedural commands that the system provides

GENERAL LEDGER RECORD

[Account Code] _____

[Account Name] _____

[Report Item] _ (Y/N)

[Debit] _____

[Credit] _____

```
Attribute summary of Data Base GLEDGER

1.Account Code:  N,4,-2147483647,2147483647,"                    "
2.Account Name: AN,36,0,36,"new account     "
3.Report Item: AN,0,0,1,"                    "
4.Debit:  N,4,-2147483647,2147483647,"000000.00        "
5.Credit:  N,4,-2147483647,2147483647,"000000.00        "

Record Size (Bytes) = 49
Total Records = 0
```

Code	Name	Balance?	Debit	Credit
401	ART AND CRAFT	Y	.00	.00
402	EDUCATION	Y	.00	.00
403	ENGLISH	Y	7.30	.00
404	MATHEMATICS	Y	1.22	.00
407	MUSIC	Y	.00	.00
408	PHYSICAL EDUCATION	Y	1.05	.00
409	SCIENCE	Y	.00	.00
410	SOCIAL SCIENCE	Y	.00	.00
416	GRADUATE GROUP	Y	.00	.00
501	ART EDUCATION	Y	.00	.00
502	PROFESSIONAL STUDIES	Y	.00	.00
503	COMMERCE	Y	.00	.00
504	MULTICULTURAL EDUCATION	Y	.00	.00
505	COMPUTER STUDIES	Y	.00	.00
506	EDUCATION	Y	3.90	.00
507	ENGLISH	Y	2.70	.00
508	DRAMA	Y	.00	.00
509	HOME ECONOMICS	Y	.00	.00
511	LANGUAGES	Y	2.34	.00
512	MATHEMATICS	Y	.00	.00
513	MUSIC	Y	12.05	.00
514	CLASSICAL STUDIES	Y	.00	.00
515	PHYSICAL EDUCATION	Y	1.44	.00
516	SCIENCE	Y	.00	.00
517	SOCIAL SCIENCES	Y	1.50	.00
518	OUTDOOR EDUCATION	Y	.00	.00
519	TECHNICAL AND CRAFT EDUCATION	Y	.00	.00
520	MAORI STUDIES	Y	.00	.00
521	TEACHING CHILDREN WITH SPECIAL NEEDS	Y	.00	.00
522	AGRICULTURE AND HORTICULTURE	Y	.00	.00
702	LIBRARY	Y	.00	.00
2303	AUDIO AND VIDEO	Y	.00	.00
2401	EDUCATION OF DEAF	Y	2.43	.00
2406	ADVISORS ON DEAF CHILDREN	Y	.00	.00
2503	DIVISION U	Y	.00	.00
2709	PUBLICATIONS COMMITTEE	Y	.00	.00
2710	CONTINUING EDUCATION	Y	.00	.00
3001	LECTURERS ADMINISTRATION	Y	4.52	.00
3002	REGISTRY	Y	3.00	.00
3003	SPECIAL COURSES	Y	.00	.00
3004	TEACHING PRACTICE (PY)	Y	.00	.00
3005	TEACHING PRACTICE (SY)	Y	.40	.00
3501	CONTROL ACCOUNT	Y	.00	44.52

	Debit	Credit
Total	44.52	44.52

Record Count = 43

Select from the following options:

1.	EXIT to Printery Menu	[Run RTNPTRN]
2.	ENTER Printery transactions	[Run TFLINT]
3.	MODIFY (unposted) transactions	[Run TFLMOD]
4.	DELETE (unposted) transactions	[Run TFLDEL]
5.	PRINT the (unposted) transactions	[Run TFLPRN]
6.	POST the transactions to the general ledger file and append to TFLMAST	[Run TFLPOST]
7.	PRINT master transaction file in order of entry	[Run TFLMSPRN]
8.	PRINT master transaction file in ledger code order	[Run TFLACPRN]

```
C*****************************************************************
C*  THIS PROGRAM WILL PERFORM THE FOLLOWING FUNCTIONS            *
C*                                                               *
C*  - BEFORE UPDATING, SAVE THE OLD VERSIONS OF THE TWO DATA     *
C*    FILES, GLEDGER AND TRANSACT                                *
C*  - VALIDATE THE TRANSACTION FILE AND LIST ANY INVALID         *
C*    TRANSACTIONS FOUND                                         *
C*  - POST THE TRANSACTION AMOUNTS TO THE GENERAL LEDGER         *
C*  - APPEND THE NEW POSTED TRANSACTIONS TO THE MASTER FILE      *
C*****************************************************************

F G,F-TEMP-GLEDGER
F T,F-TEMP-TRANSACT
F GLEDGER BY CODE
HE 'Christchurch Teachers College --- Transaction Exception Listing ',DATE
HE '-------------------------------------------------------------------
HL S,G
HE '          *******     Illegal Account codes     *******
HE '-------------------------------------------------------------------
TAKE TRANSACT GLEDGER NOT MATCHING CODE
NT RESULT BY TYPE DATE CODE NUMBER DETAILS DEBIT CREDIT

HE S,G
HE '          *******    Illegal Debit or Credit    *******
HE '-------------------------------------------------------------------
ECT TRANSACT WHERE DEBIT LT 0.0 OR CREDIT LT 0.0
NT RESULT BY TYPE DATE CODE NUMBER DETAILS DEBIT CREDIT

HE S,G
HE '          *******   Illegal Transaction Type   *******
HE '-------------------------------------------------------------------
ECT TRANSACT WHERE TYPE NE P AND TYPE NE R AND TYPE NE J
NT RESULT BY TYPE DATE CODE NUMBER DETAILS DEBIT CREDIT

SPACE *****************************************************************
SPACE *  POST THE VALID TRASACTIONS TO THE GENERAL LEDGER. ADD THE VALID *
SPACE *  TRANSACTIONS TO THE MASTER FILE AND EMPTY THE NOW REDUNDANT     *
SPACE *  TRANSACTION FILE THAT HAS BEEN POSTED (TRANSACT)                *
SPACE *****************************************************************

TAKE TRANSACT GLEDGER MATCHING CODE
ECT  RESULT WHERE DEBIT GE 0.0 AND CREDIT GE 0.0 AND TYPE IS P,R,J
NTO  TRANSACT RESULT
T    GLEDGER RESULT BY CODE AND ADD DEBIT,CREDIT
TY   TRANSACT
     TRANS
```

Appendix 4

The DBASE II Application Development System

The following is a sample of the system that was developed using DBASE II. Included is :

* The structure of the General Ledger File as kept in the dictionary

* A listing of the General Ledger using the report generator. Time taken to develop this report was in the order of 10 minutes. The time required if using a procedural language such as COBOL would be in the order of half a day. Thus report generators are a highly productive part of an Application Development System

* A listing of the Transaction File

* A typical command file consisting of both procedural and non procedural commands

* The parameters that are used to drive the report generator.

. use slbdscr
. display structure
STRUCTURE FOR FILE: SLBDSCR.DBF
NUMBER OF RECORDS: 00045
DATE OF LAST UPDATE: 07/12/88
PRIMARY USE DATABASE
FLD	NAME	TYPE	WIDTH	DEC
001	CODE	N	004	
002	NAME	C	036	
003	REPUNT	C	001	
004	BALANCE	N	010	002
** TOTAL **			00052	

.

Christchurch Teachers College -- Chart of Accounts Listing

Code	Account Name	S Y S T	Debit	Credit
401	ART AND CRAFT	Y	0.00	0.00
402	EDUCATION	Y	0.00	0.00
403	ENGLISH	Y	0.00	0.00
406	MATHEMATICS	Y	0.00	0.00
407	MUSIC	Y	0.00	0.00
408	PHYSICAL EDUCATION	Y	0.00	0.00
409	SCIENCE	Y	0.00	0.00
410	SOCIAL SCIENCE	Y	0.00	0.00
416	GRADUATE GROUP	Y	0.00	0.00
501	ART EDUCATION	Y	0.00	0.00
502	PROFESSIONAL STUDIES	Y	0.00	0.00
503	COMMERCE	Y	0.00	0.00
504	MULTICULTURAL EDUCATION	Y	0.00	0.00
505	COMPUTER STUDIES	Y	0.00	0.00
506	EDUCATION	Y	0.00	0.00
507	ENGLISH	Y	0.00	0.00
508	DRAMA	Y	0.00	0.00
509	HOME ECONOMICS	Y	0.00	0.00
511	LANGUAGES	Y	0.00	0.00
512	MATHEMATICS	Y	0.00	0.00
513	MUSIC	Y	0.00	0.00
514	CLASSICAL STUDIES	Y	0.00	0.00
515	PHYSICAL EDUCATION	Y	0.00	0.00
516	SCIENCE	Y	0.00	0.00
517	SOCIAL SCIENCES	Y	0.00	0.00
518	OUTDOOR EDUCATION	Y	0.00	0.00
519	TECHNICAL AND CRAFT EDUCATION	Y	0.00	0.00
520	MAORI STUDIES	Y	0.00	0.00
521	TEACHING CHILDREN WITH SPECIAL NEEDS	Y	0.00	0.00
522	AGRICULTURE AND HORTICULTURE	Y	0.00	0.00
702	LIBRARY	Y	0.00	0.00
2303	AUDIO AND VIDEO	Y	0.00	0.00
2401	EDUCATION OF DEAF	Y	0.00	0.00
2405	ADVISORS ON DEAF CHILDREN	Y	0.00	0.00
2503	DIVISION U	Y	0.00	0.00
2709	PUBLICATIONS COMMITTEE	Y	0.00	0.00
2720	CONTINUING EDUCATION	Y	0.00	0.00
3001	LECTURERS ADMINISTRATION	Y	0.00	0.00
3002	REGISTRY	Y	0.00	0.00
3003	SPECIAL COURSES	Y	0.00	0.00
3004	TEACHING PRACTICE (PY)	Y	0.00	0.00
3005	TEACHING PRACTICE (SY)	Y	0.00	0.00
3501	CONTROL ACCOUNT	Y	0.00	0.00
** TOTAL **			0.00	0.00

. CET PRINT OFF

Christchurch Teachers College -- Transaction Listing

Date	Code	Number	Details	Debit	Credit
03/08/82	405	737	R.HAYDEN	4.50	0.00
06/08/82	405	650	M.A.LEE	1.80	0.00
05/08/82	405	668	J.MONAN	0.72	0.00
02/08/82	405	607	J.HARVEST	0.20	0.00
03/08/82	406	663	J.LAURIE	0.72	0.00
06/08/82	406	657	R.MCPHERSON	1.20	0.00
02/08/82	409	680	P.KENNAHAN	1.05	0.00
06/08/82	506	657	D.W.F.BROWN	2.40	0.00
06/08/82	506	660	D.W.F.BROWN	0.75	0.00
06/08/82	506	661	D.W.F.BROWN	0.30	0.00
06/08/82	506	662	D.W.F.BROWN	0.45	0.00
09/08/82	507	675	R.GRAHAM	2.70	0.00
06/08/82	511	652	CLARE CLARK	2.34	0.00
06/08/82	513	653	F.E.DENNIS	3.10	0.00
06/08/82	513	665	F.E.DENNIS	2.70	0.00
09/08/82	513	686	K.POWELL	1.25	0.00
09/08/82	515	576	J.MCCARTHY	1.44	0.00
03/08/82	517	664	K.NICHOL	1.50	0.00
06/08/82	2401	669	M.MCLAGHAN [POST AGAIN]	1.50	0.00
02/09/82	2401	682	M.PARSONS	0.90	0.00
06/08/82	2001	655	GRAHAM R.	0.12	0.00
06/08/82	2001	656	R.H.MURRAY	1.30	0.00
06/08/82	3001	672	R.H.MURRAY	0.40	0.00
09/08/82	3001	674	J.F.MANN	2.50	0.00
06/08/82	3002	670	M.SHANKLAND	1.00	0.00
09/08/82	3002	683	J.MCGREGOR	2.00	0.00
06/08/82	3005	650	L.MCLURDY	0.40	0.00
03/08/82	3501	0	REG. NO.S 650-670 & 737	0.00	32.49
09/08/82	3501	0	REG. NO.S 674-687	0.00	12.12
** TOTAL **				44.52	44.61

```
>type tflmnt.cmd
*****************************************************************************
* COMMAND MODULE: TFLMNT --- MODIFY CERTAIN FEILDS OF THE TRANSACCTIONS      *
*****************************************************************************
* This program allows the user to update certain fields in the TRANSACT data *
* base records. Access to records in the data base is defined by the actual  *
* physical position.                                                         *
*****************************************************************************
SET TALK OFF
USE TRANSACT
SAVE TO Memfile
STORE T TO TFLMNT
STORE '     ' TO Trans:no
STORE '          ' TO Tmp:Date
STORE '     ' TO Tmp:Number
STORE 'xxxxxxxxxxxxxxxxxxxxx' TO Tmp:Detail
*
* Locate position of last record and save in Tfl:Last
*
GO BOTTOM
STORE STR(#,5) TO Tfl:Last
DO WHILE TFLMNT
   ERASE
   @ 0,10  SAY 'Christchurch Teachers College --- Transactiom Maintenance'
   @ 1,1   SAY '=========================================================='
           +'=========='
   @ 4,5   SAY 'Enter Record number ' GET Trans:no
   @ 4,35  SAY '(null value to exit)'
   READ
   IF &Trans:no = 0.
      RELEASE ALL.
      RESTORE FROM Memfile
      RETURN
   ENDIF
   IF &Trans:No > &Tfl:Last
      @ 21,1 SAY ' '
      ACCEPT 'Illegal record number --- Type [Return] to continue ' TO Hold:flag
   ELSE
      GOTO &Trans:No
      STORE Date TO Tmp:Date
      STORE STR(Number,5) TO Tmp:Number
      STORE Details TO Tmp:Detail
      @ 7,5   SAY 'TRANSACTION DETAILS:'
      @ 9,9   SAY 'Transaction Type '  + Type
      @ 10,9  SAY 'Transaction Code '  + STR(Code,4)
      @ 11,9  SAY 'Debit             ' + STR(Debit,10,2)
      @ 12,9  SAY 'Credit            ' + STR(Credit,10,2)
      @ 14,5  SAY 'MODIFY THE FOLLOWING --- Type [Return] To stay the same'
      @ 16,9  SAY 'Transaction Date '  GET Tmp:Date
      @ 17,9  SAY 'Transaction Nmbr '  GET Tmp:Number
      @ 18,9  SAY 'Details           ' GET Tmp:Detail
      READ
      @ 19,9  SAY ' '
      ACCEPT 'Is data entered correctly ? (Y/N) ' TO Action
      IF !(Action) = 'Y'
         REPLACE Date WITH Tmp:Date, Number WITH &Tmp:Number,;
                 Details WITH Tmp:Detail
      ELSE
         @ 21,0 SAY ' '
         ACCEPT 'Invalid Data --- Type any KEY to continue ' TO Hold:it
      ENDIF 1
   ENDIF 2
   STORE T TO Tflmnt
ENDDO Tflmnt
```

```
>type *.frm
W=80 FIRST : TRIAL    PRT : TRIAL.PRT
B



>type glist.frm
W=80
Y
Christchurch Teachers College -- Chart of Accounts Listing
N
Y
N
4,CODE
Code
N
36,NAME
Account Name
1,REPORT
Rprt
10,DEBIT
Debit
Y
10,CREDIT
Credit
Y

B



>type tflprn.frm
W=80
Y
Christchurch Teachers College -- Transaction Listing
n
Y
n
8,DATE
Date
4,CODE
Code
n
6,NUMBER
Number
n
24,DETAILS
Details
10,DEBIT
Debit
Y
10,CREDIT
Credit
Y

B
```

The FORMULA Application Development System

The following set of listings are a segment of the Printery System using this tool.

* The fixed layout of the form used for entering records into the General Ledger
* The definition of the menu option that allows records to be entered or updated for the General Ledger
* The menu option used to print out the chart of accounts listing
* A chart of accounts listing that was developed by the report generator developed using this system
* The format of the form used to enter transactions into the Transaction file
* A listing of the transactions in the Transaction file

Christchurch Teachers College
General Ledger

(9/21/82- 25 - 0)

1. Account Code..... 401
2. Account name..... ART AND CRAFT
3. Report..... Y
4. Debit..... 0.00
5. Credit..... 0.00

DMA General Accounting System
Master Menu

```
Transaction #      1
( 9/21/82-    24 --  0)
 1.  Menu item description..... Enter/Update General Ledger
 2.  Menu item number.......... 1
 3.  Step 1 info: Program name..... ENTER
 4.  .....Parameter name..... GLEDGER
 5.  TYPE(1-6:MAS,XAC,IDX,UPD,REP,EX)1
 6.  Drive(0=logged,1=A,2=B,etc.)... 1
 7.  Step 2 info: Program name.....
 8.  .....Parameter name.....
 9.  .........Type.............. 0
10.  .........Drive.............. 0
11.  Step 3 info: Program name.....
12.  .....Parameter name.....
13.  .........Type.............. 0
14.  .........Drive.............. 0
15.  Step 4 info: Program name.....
16.  .....Parameter name.....
17.  .........Type.............. 0
```

DMA General Accounting System
Master Menu

Transaction # 2
(9/21/82- 22 - 0)

```
 1.  Menu item description..... Print Chart of Accounts
 2.  Menu item number.......... 2
 3.  Step 1 info: Program name..... REPORT
 4.  .....Parameter name..... GLFLST
 5.  TYPE(1-6:MAS,XAC,IDX,UPD,REP,EX)5
 6.  Drive(0=logged,1=A,2=B,etc.)... 1
 7.  Step 2 info: Program name.....
 8.  .....Parameter name.....
 9.  .....Type..... 0
10.  .....Drive..... 0
11.  Step 3 info: Program name.....
12.  .....Parameter name.....
13.  .....Type..... 0
14.  .....Drive..... 0
15.  Step 4 info: Program name.....
16.  .....Parameter name.....
17.  .....Type..... 0
```

Christchurch Teachers College
Chart of Accounts Listing

Code	Account name	R	Debit	Credit
401	ART AND CRAFT	Y		
403	EDUCATION	Y		
405	ENGLISH	Y	7.02	
406	MATHEMATICS	Y	1.92	
407	MUSIC	Y		
408	PHYSICAL EDUCATION	Y		
409	SCIENCE	Y		
410	SOCIAL SCIENCE	Y		
416	GRADUATE GROUP	Y		
501	ART EDUCATION	Y		
502	PROFESSIONAL STUDIES	Y		
503	COMMERCE	Y		
504	MULTICULTURAL EDUCATION	Y		
505	COMPUTER STUDIES	Y		
506	EDUCATION	Y	3.90	
507	ENGLISH	Y		
508	DRAMA	Y		
509	HOME ECONOMICS	Y		
511	LANGUAGES	Y	2.34	
512	MATHEMATICS	Y		
513	MUSIC	Y	10.80	
514	CLASSICAL STUDIES	Y		
515	PHYSICAL EDUCATION	Y		
516	SCIENCE	Y		
517	SOCILA SCIENCES	Y	1.50	
518	OUTDOOR EDUCATION	Y		
519	TECHNICAL AND CRAFT EDUCATION	Y		
520	MAORI STUDIES	Y		
521	TEACHING CHILDREN WITH SPECIAL NEEDS	Y		
522	AGRICULTURE AND HORTICULTURE	Y		
702	LIBRARY	Y		
2303	AUDIO AND VIDEO	Y		
2401	EDUCATION OF DEAF	Y		
2405	ADVISORS ON DEAF CHILDREN	Y		
2503	DIVISION U	Y		
2709	PUBLICATIONS COMMITTEE	Y		
2720	CONTINUING EDUCATION	Y		
3001	LECTURERS ADMINISTRATION	Y	2.02	
3002	REGISTRY	Y	1.00	
3003	SPECIAL COURSES	Y		
3004	TEACHING PRACTICE (PY)	Y		
3005	TEACHING PRACTICE (PY)	Y	.40	
3501	CONTROL ACCOUNT	Y		30.90
43		3	30.90	30.90

Christchurch Teachers College
Transaction File

Transaction # 1
(9/21/82- 90 - 0)

1. TYPE..... R
2. DATE..... 09/08/82
3. CODE..... 9999
4. NUMBER..... 657
5. DETAILS..... D.W.F.BROWN
6. DEBIT..... 8.00
7. CREDIT..... 0.00

Christchurch Teachers College
Transaction Listing

T	Date	Code	Numbe	Details	Debit	Credit
P	08/08/82	405	737	R.HAYDEN	4.50	0.00
P	08/08/82	405	650	M.A.LEE	1.80	0.00
P	08/08/82	511	652	CLARE CLARK	2.34	0.00
P	08/08/82	513	653	F.E.DENNIS	8.10	0.00
P	08/08/82	3001	655	GRAHAM R.	.12	0.00
P	08/08/82	3001	656	R.H.MURRAY	1.30	0.00
P	08/08/82	506	657	D.W.F.BROWN	2.40	0.00
P	08/08/82	3005	658	L.MCCURDY	.40	0.00
P	08/08/82	506	660	D.W.F.BROWN	.75	0.00
P	08/08/82	506	661	D.W.F.BROWN	.30	0.00
P	08/08/82	506	662	D.W.F.BROWN	.45	0.00
P	08/08/82	406	663	J.LAURIE	.72	0.00
P	08/08/82	517	664	K.NICHOL	1.50	0.00
P	08/08/82	513	665	F.E.DENNIS	2.70	0.00
P	08/08/82	406	667	R.MCPHERSON	1.20	0.00
P	08/08/82	405	668	J.DORAN	.72	0.00
P	08/08/82	3002	670	N.SHANKLAND	1.00	0.00
P	08/08/82	3001	672	R.H.MURRAY	.60	0.00
P	08/08/82	3501	0	REG. NO.S 650-672 & 737		30.90
=	========	====	=====	=======================	==========	==========
					30.90	30.90